File Copy

BRL MR 2199

BRL

AD746957

MEMORANDUM REPORT NO. 2199

SYMAP2 - AN OPERATIONAL COMPUTER-BASED ALGEBRAIC SYMBOL MANIPULATOR

by

George C. Francis

June 1972

U.S. ARMY ABERDEEN RESEARCH AND DEVELOPMENT CENTER
BALLISTIC RESEARCH LABORATORIES
ABERDEEN PROVING GROUND, MARYLAND

B A L L I S T I C   R E S E A R C H   L A B O R A T O R I E S

MEMORANDUM REPORT NO. 2199

JUNE 1972

SYMAP2 - AN OPERATIONAL COMPUTER-BASED ALGEBRAIC SYMBOL MANIPULATOR

George C. Francis

Applied Mathematics Division

RDT&E Project No. 1T061102A14B

A B E R D E E N   P R O V I N G   G R O U N D,   M A R Y L A N D

B A L L I S T I C   R E S E A R C H   L A B O R A T O R I E S

MEMORANDUM REPORT NO. 2199

George C. Francis/ngh
Aberdeen Proving Ground, Md.
June 1972

SYMAP2 - AN OPERATIONAL COMPUTER-BASED ALGEBRAIC SYMBOL MANIPULATOR

ABSTRACT

SYMAP2, a FORTRAN computer program for symbolic manipulation of algebraic forms, is operational on the BRLESC 2 computer. FORTRAN-like formulas can be built up, combined, displayed, differentiated, modified through substitution or change of variables, and operated on in various other ways, either to verify or eliminate the need for many tedious and lengthy hand transformations. No knowledge of FORTRAN programming as such is required for use of SYMAP2. This report includes numerous examples with detailed explanation of each.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. BACKGROUND

Traditionally the electronic computing machine has been used as a large, fast, accurate device for numerical calculations where either many steps, many decimal places of accuracy, or complicated logic as to what steps to take next are involved. In recent years there have been a number of attempts to extend the speed, the capacity, and the logic of the computer to a variety of non-numeric or partially non-numeric applications[1]. The computer-based algebraic symbol manipulator is one such extension.

A mathematician, or other scientist or engineer, using pencil and paper often has to carry out tedious algebraic and related symbol manipulations in transforming mathematical expressions to more useful forms. Obvious examples include expanding powers of sums, grouping terms having some common factor, performing differentiation with respect to one or more variables, changing coordinate systems, and the like. In the process it is easy to make careless errors such as losing a sign, overlooking one term, or depending on faulty recollection of some mathematical rule. Electronic computers can be programmed to carry out many symbolic transformations quickly and accurately and thus save the mathematician both time and effort as well as reduce the likelihood of a careless error. SYMAP2 was designed to do just that and is now operational at BRL.*

Earlier attempts to achieve some of these goals have been made both at BRL and elsewhere. P. Smith[2] successfully attacked the differentiation problem at BRL using a Polish suffix notation internally. Very little use of his programs has been made locally. While these programs still exist within ARDC,** they are programmed in the FORAST language rather than the more standard FORTRAN IV. SYMAP1 is a more recent algebraic symbol manipulator designed at BRL by the author[3], and it is available in FORTRAN IV. It is largely restricted to manipulations of polynomials, although decimal exponents are permitted and some trigonometric functions are allowed as coefficients. Differentiation of polynomials in several variables is included, and

* *Ballistic Research Laboratories*
***Aberdeen Research & Development Center*

a variety of substitutions can be performed. SYMAP2, the subject of this report, includes all these capabilities and many more.

Other algebraic symbol manipulators have been devised elsewhere. Brief discussions of some of these are to be found in surveys by Bobrow[4] and Sammet[5], which contain extensive bibliographies. Perhaps the best known of these are FORMAC[6] and ALTRAN[7]. None of these outside manipulators is available on computers at BRL since most of them were designed for other specific computers. Brief comparisons with SYMAP2 will appear occasionally later in this report.

II. THE MASAITIS L-SYSTEM AND SYMAP 2

C. Masaitis of the Applied Mathematics Division of BRL has developed an original approach to the algebraic symbol manipulation problem. It is called the L-System and is described in detail in an as yet unpublished manuscript[8]. A few comments are in order here, as SYMAP 2 is an implementation by the author and Miss V. Woodward of AMD of much of the L-System with minor changes for operational expediency. The implementation in turn influenced certain alternative design characteristics, and the L-System was modified to reflect some of these.

A special internal notation (L-notation) was developed by Masaitis for mathematical forms in the L-System. This notation, while better read by machines than by humans, removes the inherent ambiguity of some standard mathematical conventions and makes precise many important relationships between component parts of the mathematical forms no matter how complicated the forms. Given this notation and Masaitis' related precise rules of manipulation, the computer program SYMAP2 could be implemented and has been within the constraints of the computers currently available at BRL. Flow charts and related details of the implementation are presented in another as yet unpublished report[9].

In general any mathematical form is permitted which can be built up from constants and primitives by repeated application of the operations of addition, subtraction, multiplication, division, exponentiation, and certain standard or general functions of one or more variables. The set of simple functions of one variable implemented at present includes $\log_e$, $\log_{10}$, sin, cos, tan, cot, sec, csc, arc sin, arc cos, arc tan, arc cot, sinh, cosh, and tanh. Others could be added if needed. The user can also specify such general functions as $f(x)$, $g(u,v)$, etc. Repeated applications of these operations can build up composite expressions of considerable complexity which in turn can be operated on further.

Additional operations include summations over one or more sets of integer indexes, products over index sets, indefinite integration*, definite integration*, differentiation, substitution of one algebraic form for another in a third, solving (small) systems of linear algebraic equations, change of variables, and several kinds of factoring. With each of these manipulations certain kinds of expansion and simplification are automatic, and others are under user control. Results can be displayed in readable form following any such operation.

Most computer problems at BRL are written in the FORTRAN language, and SYMAP2 consists of more than 120 FORTRAN subprograms interlinked appropriately to achieve the aims of the variety of manipulations mentioned above. Several versions of SYMAP2 exist, all functioning on the BRLESC 2 computer. One version in "standard" FORTRAN could be implemented elsewhere with at most minor changes. Other versions provide some additional flexibility by using certain special features of local computers and local FORTRAN conventions and subroutines. Where pertinent in this report these versions will be distinguished.

*The evaluation of integrals has been implemented, but is limited to certain very commonly occurring classes of integrands, as discussed later.

III. INPUT TO THE SYMAP2 MANIPULATOR

As with most FORTRAN programs some "data" must be supplied to the SYMAP 2 manipulator. This input consists of several distinct parts introduced sequentially:

(1) Standard input common to all users.

(2) Mode specification and list of user primitives.

(3) List of user special functions (at least one).

(4) List of user manipulations and special controls.

The present implementation of SYMAP2 at BRL is as a "batch" problem, not remote and not interactive; so the input is prepared in advance and submitted on punched cards.

A. Part (1) Input

The part (1) input is provided to the user and merely submitted first. It supplies certain standard symbols and symbol strings used internally and is read in rather than prestored to provide flexibility in case of implementation on dissimilar computers. In particular, the special primitive names PI ($=\pi$), EBASE ($=e$), and INFIN ($=\infty$) are supplied here, not in part (2). This part terminates with a "sentinel card" consisting of commas in card columns 1 and 2 only.

B. Part (2) Input

Input for part (2) is of two types, one card for certain user options and additional cards for specifying user primitives. The options include: (a) exact (rational) or approximate (decimal) arithmetic, (b) stopping if an indeterminate form is encountered ($0/0$, $0.\infty$, ∞/∞, 0^0, ∞^0, or 1^∞), or replacing it by a new primitive and continuing, and (c) specifying a control on certain automatic expansions.

If all numerical coefficients are integers, specifying either EXACT or APPROX at column 1 will suffice. Otherwise if rational numbers are wanted, EXACT should be used. If decimal values are wanted, APPROX is needed. There are limitations of about 7 digits on numerator, denominator, and decimal approximation in the BRLESC 2 implementation. Rounding errors can occur but have not been serious to date.

Rational numbers are expressed in the form (x,y) where x and y are integers. Thus 1/3 is denoted (1,3) and - 25/231 is (-25, 231). Results in the EXACT mode are reduced to lowest terms automatically, so that (5,8) + (7,8) = (3,2). If the APPROX mode is specified, then any rationals such as (1,3) introduced by the user are automatically converted to decimal values like .3333333 with some error in the last digit. Similarly a decimal number like .3333333 in the EXACT mode becomes (3333333, 10000000) and not (1,3); so the user should exercise caution in this regard.

For control of indeterminate forms use BYPASS at column 11 if indeterminates are to be allowed in the form of new primitives. Leave columns 11-20 blank if the detection of any indeterminate is to be announced but operation is to cease then.

For control of automatic expansion, specify a small non-negative integer M in columns 21-22. Then expressions such as $(A + B)^J$ will be expanded only if J is an integer such that $0 \leq J \leq M$ and left unexpanded otherwise. Expansion in full is automatic if M=0 (or blank). The user may revise his choice of options if necessary later, in part (4). A summary is shown in Figure 1.

The remainder of part (2) of the input is a list, one per card at column 1, of the primitives the user includes in his algebraic forms. Not more than 40 are allowed at present. Their sequence in the list determines a sorting (collating) sequence used in simplifying results; so if a result is to have its terms grouped in like powers of X, for example, then X should be specified before any other primitives which are expected to appear.

Primitives may have names of 1 to 6 letters or digits, the first a letter, and if the first is F or I the second must not be a digit. Thus X, Y, XI, LAMBDA, FF, G, J3, FA2 (but not F2A or I1) are allowed. Input to part (2) ends with a sentinel card as above.

Card Columns 1-10	Arithmetic mode
EXACTbbbbb	Rational arithmetic of numbers
APPROXbbbb	Decimal approximations
Card Columns 11-20	Indeterminates
bbbbbbbbbb	Halt on noting indeterminate
BYPASSbbbb	Replace indeterminate by new primitive
Card Columns 21-30	Expansion control
bbbbbbbbbb	Expand wherever possible
integer M	Expand only if exponent ≤ M

Note: The symbol b indicates a blank character.

Figure 1. Summary of Options, Part (2) Input

C. Part (3) Input

The part (3) input is a similar list of user special functions (at least one and not more than 30) followed by a sentinel card. Function names are restricted in the same manner as primitive names and must not duplicate any primitive name, of course. The number of arguments of a function is not specified here, only the name. (As above, position on this input list affects the sorting sequence of functions, but in general all functions follow the last primitive; so this is rarely important to the user.)

D. Part (4) Input

Input for part (4) is the sequence of user-specified manipulations and controls (explained in the next section) followed by a sentinel card as above. In general there is no further input after part (4), and the manipulations are carried out in sequence when this final sentinel is recognized. Provision to restart at part (2), or at part (4), can be arranged if needed.

IV. SPECIFYING MANIPULATIONS

Manipulations allowed with SYMAP2 are of three kinds: basic, composite, and control. Basic manipulations are allowed in all implementations. Composite and control types require special features available at BRL but not necessarily elsewhere (such as the PACK, UNPACK, ENCODE, and DECODE operations[10] on character strings which are not standard FORTRAN).

A. Basic Manipulations

Basic manipulations are specified in a manner similar to some "three-address-code assembly languages". In general the result of any manipulation remains available for later use; so it is given a name. Names of most results at present are denoted by the letter F followed by an arbitrary positive integer assigned by the user. (Exceptions will be explained later). Thus in F21 = X + Y the result is called F21. The two operands are the (previously declared) primitives X and Y, and the operation is (symbolic) addition. Thus F21 here represents the sum x + y, and later operations on F21 would be operations on x + y. Operands can be primitives, positive numbers, or the results of previous operations. The second operand may be an explicitly negative number, but not the first. (However, unary operations with - permit the introduction of negatives in general.) The following examples introduce several basic manipulation types. The brief comments often shown at the right in examples are of course not punched on actual input cards.

F1 = 1.3 + X	addition
F2 = 6 - Y	subtraction
F3 = 3 * X	multiplication
F4 = Y / 4	division
F5 = X ** -2	exponentiation
F6 = SIN (F1)	function of one argument
F7 = - F3	change of sign (unary)
F8 = F7 * F1	operation on 2 previous results

F9 = GFCN (F2, F7, -8)	function of 3 arguments (see part (3) input)
F10 = F4 ** F5	higher level exponentiation
F11 = F2	copy (unary)
F12 = -3.1416	form a negative number

With SYMAP2, unlike conventional FORTRAN, the decimal point is not needed (but is permitted) with integers such as those shown in F2, F3, F4, and F9 above.

In most of the above examples no simplification is required, but it would be automatic in the case of F8 where F7 * F1 would mean (-3x) * (1.3 + x) and would become the equivalent of $-3.9x - 3x^2$. Similarly in F10 the quantity $(.25y)^{x^{-2}}$ would become $(.25)^{x^{-2}} * y^{x^{-2}}$ automatically in accordance with the rules of the L-System as adopted and implemented.

Additional basic manipulations include the following as examples, discussed briefly below:

```
F21 = SUM (1, 4, X, F3)
F22 = PROD (2, 4, Y, F2)
F23 = DERIV (X, F6)
F24 = IINTEG (X, F6)
F25 = DINTEG (F2, F3, T, F5)
F26 = SUBST (F6, X, F4)
F27 = COEFF (X, F8)
F28 = VALUE (F21)
F29 = VALUE (F22)
F30 = VALUE (F23)
F31 = RECONV (F9)
```

Here F21 is the algebraic form equivalent to the <u>indicated</u> sum over an integer index set, namely $\sum_{x=1}^{4} (3*x)$. The sum is <u>not</u> automatically expanded as this is often undesirable. However, F28 is the <u>value</u> of this summation, i.e., the expanded form with any "standard" simplifications carried out. Thus, F28 is 3*1 + 3*2 + 3*3 + 3*4 which simplifies to 30. Similarly, F22 is the indicated product

$\prod_{y=2}^{4} (6-y)$, not expanded. However, F29 is the expanded and simplified result, not necessarily a number in general, but in this case found to be $(6-2) * (6-3) * (6-4) = 4 * 3 * 2 = 24$. Double and triple sums and products are also allowed.

F23 is the indicated derivative with respect to x of the form F6. Its value is F30, namely the value of $\frac{d}{dx}$ [sin (1.3 + x)] or cos (1.3 + x). In some versions of SYMAP2 the DERIV operator includes the VALUE operation as a convenience to the user.

Much more complicated derivatives can be found, some of which are discussed later in this report.

F24 and F25 are indicated indefinite and definite integrals respectively. Certain kinds of symbolic integrals can be evaluated; others are merely indicated at present. There are no plans currently to attack the general symbolic integration problem, however. Here F24 is the equivalent of $\int \sin(1.3 + x)\, dx$ and F25 represents $\int_{6-y}^{3x} x^{-2}\, dt$. Indicated double and triple integrals are also provided for. Examples are included later in this report.

F26 is the result of an actual substitution in which the primitive X is replaced by F4, that is .25y, wherever X occurs in F6. Now F6 represents sin (1.3 + x); so F26 represents sin (1.3 + .25y). Examples of more complicated substitutions are given later in this report.

F27 is the coefficient of the first power of x in the form F8, namely -3.9. The coefficient in some other case might have been non-numeric, such as a function of y, etc. Finding such coefficients is one type of factoring.

F31 is a display (readable print-out) of the form F9, generated earlier. Displays <u>cannot</u> be manipulated further and indeed are not saved; so any operation on F31 would fail. F9 remains available, however, and can be manipulated further.

Other basic manipulations can be used to set up and solve small systems of linear algebraic equations with non-constant coefficients. (Constant coefficients are permitted, but more efficient methods are available for such cases, using standard FORTRAN subroutines.) Manipulations of type FSYST, SOLVEQ, and ELEM are needed in conjunction with types introduced earlier. As an example:

```
        F45 = FSYST (F41, F32, F39)
```

Assuming that F41, F32, and F39 are 3 linear expressions in 3 unknowns, this step forms an ordered system of 3 equations (each assumed equal to 0).

```
        F46 = FSYST (X, Y, Z)
```

This step forms a similar system of the 3 unknowns in the order specified.

```
        F47 = SOLVEQ (F45, F46, 3)
```

This solves the 3 equations specified at F45 for the 3 unknowns specified at F46 and forms an ordered array of the solutions at F47.

```
        F48 = ELEM (F47, 1)
        F49 = ELEM (F47, 2)
        F50 = ELEM (F47, 3)
```

These steps isolate the first, second, and third elements of the array F47 for any further operations, such as display.

```
        F51 = RECONV (F49)
```

This displays in readable form the item specified, in this case the second element of the solution, that is, the value of the variable Y in terms of the parameters other than X and Z which occur in the original system F45.

Incidentally, the "unknowns" in such systems <u>need</u> <u>not</u> be primitives so long as the equations are linear in them. For instance F41 might be $x^2 + 2y + z^2 + a \cdot \sin(w^{1/2}) + cp^v - 4$ and the unknowns could be p^v, $\sin(w^{1/2})$, and x^2. The solutions would involve y, z, a, c, and any parameters introduced via the other equations.

Similar commands could be used (in principle) for systems of 1,2, 3,4 or more elements. A limit of at most 3 elements is imposed at present on BRLESC 2.

Additional information on several of these and a few additional SYMAP2 commands is found in the examples given later in this report.

A brief summary is shown in Figure 2(a) and (b) below:

Arithmetic

E1 + E2	E1 * E2
E1 - E2	E1 / E2
-E1	E1 ** E2

Simple Functions

LOGE (E1)	ARCSIN (E2)
LOG10 (E1)	ARCCOS (E2)
SIN (E2)	ARCTAN (E2)
COS (E2)	ARCCOT (E2)
TAN (E2)	SINH (E2)
COT (E2)	COSH (E2)
SEC (E2)	TANH (E2)
CSC (E2)	

Here Ei is a primitive, a positive constant, or the label F_j of a previous result. E2 but not E1 can be an explicitly negative constant also. E1 cannot be a constant. i, j, and k are positive integers, and p is a primitive. See also Figure 2(b).

Figure 2(a). Summary of Principal Basic Manipulation Types

B. Composite Manipulations

Composite manipulations allow for the equivalent of several basic manipulations in one user specification. Thus algebraic formulas much like the "arithmetic expressions" of FORTRAN are permitted:

```
F55 = (Y**2 - X**(Y-1)) * (SIN (F3 + Y * LOGE (X**(Y-1))))**2
```

Use of such a formula may make specification simpler for users familiar with FORTRAN expressions. The SYMAP2 program analyzes such expressions, breaks them up into equivalent basic manipulations and in due course carries out those basic manipulations. Advantages are that fewer cards are needed, and meaningful formulas are kept intact by the user. Disadvantages are that the manipulator program is enlarged and is slowed down by the time needed to analyze complicated expressions, and further that some basic manipulations may be done more often than

Special Functions

RECONV (F_j)

VALUE (F_j)

COEFF $(\overline{E1}, F_j)$

FSYST $(\overline{E1}, \overline{E1}', \ldots, \overline{E1}'')$

SOLVEQ (F_j, F_j', i)

ELEM (F_j, i)

SUBST $(F_j, \overline{E1}, E2)$

DERIV (p, F_j)

IINTEG (p, F_j)

IINTEG (p, p', F_j)

IINTEG (p, p', p'', F_j)

DINTEG $(E2, E2', p, F_j)$

DINTEG $(E2, E2', p, E2'', E2''', p', F_j)$

DINTEG $(E2, E2', p, E2'', E2''', p', E2'''', E2''''', p'', F_j)$

SUM (i, k, p, F_j)

SUM $(i, k, p, i', k', p', F_j)$

SUM $(i, k, p, i', k', p', i'', k'', p'', F_j)$

PROD (i, k, p, F_j)

PROD $(i, k, p, i', k', p', F_j)$

PROD $(i, k, p, i', k', p', i'', k'', p'', F_j)$

SEQ $(E2, E2', \ldots, E2'')$

IMDIFF (p, F_j, p', F_j')

IMDIFF $(p, F_j, p', F_j', p'', F_j'')$

IMDIFF $(p, F_j, p', F_j', p'', F_j'', p''', F_j''')$

PIMDIF $(i, i', F_j, F_j', F_j'', F_j''')$

CHGVAR $(i, i', i'', F_j, F_j', F_j'', F_j''', F_j'''')$

Figure 2(b). Summary of Principal Basic Manipulation Types necessary. In F55 for example X**(Y-1) appears twice and would be generated twice. However, the user could generate it once, as F54 say, and refer to F54 twice in specifying F55.

Certain special basic manipulations such as ELEM, VALUE, RECONV, COEFF, etc. are not permitted in composite manipulations per se but can be done separately and the results referred to by label. A length limitation to card columns 1 - 70 is also imposed on both basic and composite manipulations at present.

C. Control Manipulations

Control specifications allow certain kinds of counting, decisions based on counts, and jumps. Some use of indexed arrays is also permitted. Jumps are to numbered manipulations only. At present up to 100 different cards may have unique one or two digit numbers in columns 79 - 80. The unconditional jump, say to the manipulation numbered 40, is as follows:

```
GO TO 40
```

where the GO starts at card column 1. This causes the manipulation numbered 40 to be done next, then the one after 40, etc. until some other control specification changes the sequence again.

Other jumps depend on the integer contents of certain pseudo-index registers called I1, I2, ..., I20. At present there are precisely 20 of these. Their existence rules out primitives and functions having names starting with the letter I followed by a digit, as stated earlier. Each index register I_n can contain one integer, positive, negative, or zero. Loading an index register, say I3, with an integer, say 2, is done with the following control specification, starting at card column 1:

```
INDEX I3 = 2
```

The integers in index registers may be changed by adding, subtracting, multiplying, or dividing by constants or by the contents of other index registers. (The integer result of division is "rounded down" the same as in FORTRAN). Thus

```
INDEX I2 = I2 + 1
INDEX I2 = I3 - 5
INDEX I4 = 3 * I1
INDEX I4 = I1 * 5
INDEX I5 = I3 / 4
INDEX I3 = I1 - I2
```

and similar integer operations using two operands are allowed.

The only conditional jump allowed at present is similar to the "Jump if +" operation in some computer codes. In SYMAP2 it takes the form

IF (I_n , m)

where I_n is any index register and m represents any numbered manipulation. Thus

```
IF (I3, 25)
```

tests the integer contents of I3 and if zero or greater causes a jump to the manipulation numbered 25.

Use of the above GO TO, INDEX, and IF types of control specifications permits many of the logical and looping capabilities of numeric processors. These are operational on the BRLESC 2 computer. Other types may be added to the SYMAP2 manipulator in the future.

It should be noted that the manipulations

```
INDEX I5 = 1
```

and

```
F3 = 1
```

are not equivalent. The first puts the integer 1 into a single cell called I5 which can be used as just described. The second puts several characters, equivalent to unity in the L-notation, into a string called F3. Conversion between the two notations is permitted, however, by additional index operations. Thus:

```
INDEX F7 = I4
```

causes the integer at I4 to be transformed into L-notation and stored in string F7 (with I4 not changed), and

INDEX I3 = F2

causes the L-integer in string F2 to be converted to a true integer and stored in index I3 (with F2 unchanged). The latter requires that F2 actually be an integer in L-notation, not some other algebraic form, or an error occurs.

When SYMAP2 control specifications are available (as on BRLESC 2), certain indexed arrays of results are also permitted. Names of up to 9 characters such as F1(2), F2(3,5), F3(I2), F4(I1,5), F6(I2,I3), F7(2,3,4) are allowed both as results and as arguments for later manipulations. Only indexes I1 through I9 are permitted in arrays at present. Note that a name like F35(I1,I3) exceeds the 9 character limitation, as does F1(I1,I2,I3). Actually, names with references to index registers I_n are adjusted internally, with the name I_n being replaced by the contents of I_n. Thus, if I1 contains 100, then the name F40(I1,2), which has 9 characters, is changed to F40(100,2), which has 10 characters and is illegal. The contents of I1 must be kept small to avoid this. In practice, however, SYMAP2 arrays must be kept small for other reasons; so this naming restriction is not serious.

Result names F_i and of course I_n can be reused if desired. In such cases previous results with exactly the same name are no longer available.

V. EXPOSITORY EXAMPLES

The principal manipulations of basic, composite, and control types have now been introduced. Examples follow to show additional features of some of these and to indicate how they can be applied in practice. Some comments on generality are included with each and also in Section VI.

A. Indexed Arrays and Controls

Given two 3 x 2 matrices A and B, find C = A + B.

Assume that the elements of A have been found and are the symbol strings with names F1(1,1), F1(1,2), F1(2,1), F1(2,2), F1(3,1), and F1(3,2) and that those of B are in a similar array F2(i,j). Let us find the elements of C and store them in an array F3(i,j).

For this specific case 6 steps are sufficient:

```
F3(1,1) = F1(1,1) + F2(1,1)
F3(1,2) = F1(1,2) + F2(1,2)
F3(2,1) = F1(2,1) + F2(2,1)
F3(2,2) = F1(2,2) + F2(2,2)
F3(3,1) = F1(3,1) + F2(3,1)
F3(3,2) = F1(3,2) + F2(3,2)
```

Six more steps of type RECONV are needed if the results are to be displayed. (See Figures 3 and 4).

If, however, the number of rows or columns were much larger, say 6 x 8, or perhaps changing for different applications, use of indexes and loops might avoid duplicate human effort.

```
INDEX I5 = 6                                no. of rows
INDEX I6 = 8                                no. of columns
INDEX I1 = 1                                first row
INDEX I2 = 1                           10   first column
F3(I1,I2) = F1(I1,I2) + F2(I1,I2)      20   one element
F100 = RECONV (F3(I1,I2))                   display
INDEX I2 =I2  + 1
INDEX I3 = I6 - I2                          last column?
IF (I3,20)

INDEX I1 = I1 + 1
INDEX I4 = I5 - I1                          last row?
IF (I4,10)

INDEX I1 = 1                                dummy operation
,,                                          (or next step, if any)
```

```
APPROX                                          options
X                                               primitive(s)
Y
,,
G                                               function(s)
,,
F1(1,1) = Y*SIN(X) + Y ** - 3                     a11
F2(1,1) = SIN(X) * 3*Y-Y ** 2.5                   b11
F1(1,2) = X * SIN(X) ** 2                         a12
F2(1,2) = X * COS(X) ** 2                         b12
F1(2,1) = (5,6) * G (Y)
F2(2,1) = (1,3) * G (Y)
F1(2,2) = .4 * Y ** 3
F2(2,2) = (Y * -.2) ** 3
F1(3,1) = (X + Y) ** 3
F2(3,1) = (X - Y) ** 3
F1(3,2) = X - Y                                   a32
F2(3,2) = Y - X                                   b32
F3(1,1) = F1(1,1) + F2(1,1)                       c11
F3(1,2) = F1(1,2) + F2(1,2)                       c12
F3(2,1) = F1(2,1) + F2(2,1)                       c21
F3(2,2) = F1(2,2) + F2(2,2)                       c22
F3(3,1) = F1(3,1) + F2(3,1)                       c31
F3(3,2) = F1(3,2) + F2(3,2)                       c32
F100 = RECONV (F3(1,1))                         display(s)
F200 = RECONV (F3(1,2))
F300 = RECONV (F3(2,1))
F400 = RECONV (F3(2,2))
F500 = RECONV (F3(3,1))
F600 = RECONV (F3(3,2))
```

Figure 3. Input, Parts (2), (3), (4), for Array Example

```
F100
  4*Y*(SIN(X))+(-1)*Y**2.5+Y**(-3)
F200
  X*(SIN(X))**2+X*(COS(X))**2
F300
  1.166667*(G(Y))
F400
  .392*Y**3
F500
  6*X*Y**2+2*X**3
F600
  0
```

Figure 4. Principal Output, Array Example

Note that in the second example an inner loop starting at step 20 is traversed for each element of any given row and an outer loop starting at step 10 allows consideration of each row in turn. The results of step 20 are stored with names F3(1,1), F3(1,2), ..., F3(1,8), F3(2,1), ..., F3(6,8). Thus there is no conflict in the reuse of the "label" F3(I1,I2), and all A's, B's and C's remain available for further use.

B. Product over an Index Set

Let us use SYMAP2 to generate $\prod_{i=1}^{3} (i^2 + j)$ and then evaluate it.

The operand $i^2 + j$ can be built up in two basic steps.

F1 = I**2	i^2
F2 = F1 + J	$i^2 + j$

where I, J are previously specified primitives.

Now indicate the product for i running from 1 through 3.

F3 = PROD (1, 3, I, F2)

This single product requires one triplet 1, 3, I indicating the lower limit, the upper limit, and the running index, followed by a final

```
Input:

EXACT
I
J
,,
G
,,
F1 = I**2
F2 = F1 + J
F3 = PROD (1, 3, I, F2)
F4 = VALUE (F3)
F5 = RECONV (F4)
,,

Principal Output:

F5
  49*J+14*J**2+J**3+36
```

Figure 5. Product over a Single Index Set

parameter specifying the operand of the product. The indicated product is called F3.

F4 = VALUE (F3)

In this context VALUE substitutes successively $i = 1$, $i = 2$, and $i = 3$ in $i^2 + j$ and multiplies the results to get the equivalent of $(1^2 + j) * (2^2 + j) * (3^2 + j)$ or $(1 + j) * (4 + j) * (9 + j)$ which automatically expands to $36 + 49j + 14j^2 + j^3$ at F4.

F5 = RECONV (F4)

This displays the final results in readable form. (See Figure 5.)

Double and triple products can be handled similarly, within limits imposed by computer memory allocations.

The general single product, double product, and triple product, such as

$$\prod_{x1=\ell 1}^{u1} f \quad \text{or} \quad \prod_{x2=\ell 2}^{u2} \prod_{x1=\ell 1}^{u1} f \quad \text{or} \quad \prod_{x3=\ell 3}^{u3} \prod_{x2=\ell 2}^{u2} \prod_{x1=\ell 1}^{u1} f$$

are specified respectively by

PROD (L1, U1, X1, F)

PROD (L2, U2, X2, L1, U1, X1, F)

PROD (L3, U3, X3, L2, U2, X2, L1, U1, X1, F)

where

F is the operand, usually dependent on X1, X2, X3

L1, L2, L3 are the lower limits

U1, U2, U3 are the upper limits

X1, X2, X3 are the running indices (bound primitives)

U1-L1, U2-L2, U3-L3 are integers ≥ 0

If such a product is to be evaluated using the VALUE operator, the limits must normally be explicit integers. However, special cases such as $\ell 1 = n$, where n is a primitive, and $u1 = n + 5$ are also allowed.

The operand F may not contain indexed arrays such as F1(X1,X2) at present.

C. <u>Sum over a Double Index Set</u>

Let us use SYMAP2 to generate $\sum_{i=1}^{3} \sum_{j=2}^{3} (i^2 + j)$ and then evaluate it.

The double summation can be specified in one step, once the summand has been built up. We assume that I and J have been previously specified as primitives.

F1 = I**2	i^2
F2 = F1 + J	$i^2 + j$

The summand $i^2 + j$ is thus called F2 here.

F3 = SUM(1, 3, I, 2, 3, J, F2)

The double summation in the L-system requires seven parameters, namely two triplets for the two index sets and one summand. The first triplet 1, 3, I specifies lower limit, upper limit, and name of index for the last (outermost) summation. The second triplet 2,3, J is similar for the next (here, innermost) summation. The summand is specified as the last parameter. At this point F3 is the indicated double sum in L-notation.

F4 = VALUE (F3)

In this context VALUE causes the successive substitution of j = 2 and then j = 3 in the summand and sums the results to obtain $i^2+2 + i^2+3$ or $2i^2 + 5$. It then uses $2i^2 + 5$ as the new summand (still symbolic, note) and successively substitutes i = 1, i = 2, i = 3 and sums to get $2(1)^2+5 + 2(2)^2+5 + 2(3)^2+5$ or 7 + 13 + 23 or 43. Thus the result F4 is the L-form equivalent to the number 43.

F5 = RECONV (F4)

This displays the readable result 43. (See Figure 6.)

Naturally, if the summand had contained other variables or functions the result would have been non-numeric. Some simplification would take place but the result might not be in the "simplest" form for the user. Additional substitutions might then be applied through other manipulations, if desired.

Triple sums and of course single sums can be specified in a similar way.

The general single, double, and triple sums such as

$$\sum_{x_1=\ell_1}^{u1} f \quad \text{or} \quad \sum_{x2=\ell 2}^{u2} \sum_{x1=\ell 1}^{u1} f \quad \text{or} \quad \sum_{x3=\ell 3}^{u3} \sum_{x2=\ell 2}^{u2} \sum_{x1=\ell 1}^{u1} f$$

are specified respectively by

SUM (L1, U1, X1, F)

SUM (L2, U2, X2, L1, U1, X1, F)

SUM (L3, U3, X3, L2, U2, X2, L1, U1, X1, F)

with the same restrictions specified earlier for products over index sets. In particular note that U1-L1, U2-L2, U3-L3 must be integers ≥ 0.

Input:

```
APPROX
I
J
,,
FCN
,,
F1 = I ** 2
F2 = F1 + J
F3 = SUM (1,3,I,  2,3,J,  F2)
F4 = VALUE (F3)
F5 = RECONV (F4)
,,
```

Principal Output:

```
F5
  43
```

Figure 6. Sum over a Double Index Set

D. Substitution in Manipulating Trigonometric Identities

Given that

sin (A + B) = sin A * cos B + sin B * cos A

and that

cos (A + B) = cos A * cos B - sin A * sin B

let us derive formulas for sin 3A and for cos 3A in terms of sin A and cos A.

Assuming that composite manipulations are available (otherwise taking a few more, but equivalent, steps) we can write:

```
F1 = SIN (A) * COS (B) + SIN (B) * COS (A)
```

which is equivalent to sin (A + B), and similarly:

```
          F2 = COS (A) * COS (B) - SIN (A) * SIN (B)
```

which is equivalent to cos (A + B). We note that if B were set equal to A in F1 and F2, we would get the equivalent of sin (A + A) or sin 2A and of cos (A + A) or cos 2A. Similarly for B = 2A we would get the equivalent of sin 3A and of cos 3A in terms of functions of A and 2A.

```
          F11 = SUBST (F1, B, A)
          F12 = SUBST (F2, B, A)
          F13 = 2 * A
          F14 = SIN (F13)
          F15 = COS (F13)
```

We now have explicitly sin 2A at F14 and cos 2A at F15 as well as their equivalent expressions in terms of A at F11 and F12.

```
          F21 = SUBST (F1, B, F13)
          F22 = SUBST (F2, B, F13)
```

Now F21 is equivalent to sin 3A and F22 is equivalent to cos 3A, both in terms of A and 2A. Let us remove the dependence on 2A by use of F11 through F15.

```
          F23 = SUBST (F21, F14, F11)
```

This eliminates sin 2A in the expression for sin 3A.

```
          F24 = SUBST (F23, F15, F12)
```

This eliminates cos 2A from the result of the preceding step. Similarly,

```
          F25 = SUBST (F22, F14, F11)
          F26 = SUBST (F25, F15, F12)
```

Now the formula wanted for sin 3A is at F24 and that for cos 3A is at F26. These can be displayed with two more steps:

```
          F100 = RECONV (F24)
          F200 = RECONV (F26)
```

These results involve sin A and cos A to various powers. If some other combination is preferred, then replacing $\sin^2 A$ by $1 - \cos^2 A$ or $\cos^2 A$ by $1 - \sin^2 A$ etc. in F24 and F26 (or perhaps in F12 before continuing as shown) might give the form wanted. In general it is not

```
APPROX
A
B
,,
FCN
,,
F1   = SIN (A) * COS (B) + SIN (B) * COS (A)
F2   = COS (A) * COS (B) - SIN (A) * SIN (B)
F11  = SUBST (F1, B, A)
F12  = SUBST (F2, B, A)
F13  = 2 * A
F14  = SIN (F13)
F15  = COS (F13)
F100 = RECONV (F11)
F200 = RECONV (F12)
F300 = RECONV (F14)
F400 = RECONV (F15)
F21  = SUBST (F1, B, F13)
F22  = SUBST (F2, B, F13)
F500 = RECONV (F21)
F600 = RECONV (F22)
F23  = SUBST (F21, F14, F11)
F24  = SUBST (F23, F15, F12)
F700 = RECONV (F24)
F25  = SUBST (F22, F14, F11)
F26  = SUBST (F25, F15, F12)
F800 = RECONV (F26)
,,
```

Figure 7. Input for Substitution Example

```
F100
  2*(SIN(A))*(COS(A))
F200
  (-1)*((SIN(A))**2)+((COS(A))**2
F300
  (SIN(2*A))
F400
  (COS(2*A))
F500
  (SIN(A))*(COS(2*A))+(SIN(2*A))*(COS(A))
F600
  (-1)*(SIN(A))*(SIN(2*A))+(COS(A))*(COS(2*A))
F700
  3*(SIN(A))*((COS(A))**2)+(-1)*((SIN(A))**3)
F800
  (-3)*((SIN(A))**2)*(COS(A))+((COS(A))**3)
```

Figure 8. Principal Output, Substitution Example

obvious in advance just which substitutions of this kind are useful. Hence they can not be fully automatic. However, the user can try several versions of his "program" one after the other, and select the form of result he prefers.

Figures 7 and 8 show the input and output for an equivalent set of steps with a few additional results displayed.

Substitution in SYMAP2 is by no means limited to trigonometric expressions. Indeed, primitives, powers, products, and sums can be replaced in most contexts. In particular, a product like $a^2 \cdot x^4$ can be replaced in a context like $\log (3 \cdot a^2 \cdot b^3 \cdot x^4 \cdot y)$ where the factors are not even adjacent. Many symbol manipulators do not have this capability.

E. Indefinite and Definite Integration

Let us find the indefinite integral with respect to y and then x of a sum of terms involving powers of y and x, namely:

$$\int\int(2 x^4 y + x^{-1} y^2 \sin (t)) \, dy \, dx$$

Then let us find the following definite integral as well:

$$\int_a^h \int_{t^2}^x a^x \cos(y) \, dy \, dx$$

The two SYMAP2 procedures are analogous. In each case the integrand is set up, and then a single additional step is sufficient to specify a single, double, or even triple integral. If the limits of integration are complicated, they probably should be formed separately for convenience. Finally the results can be displayed in readable form.

```
F1 = 2 * X ** 4 * Y + X ** -1 * Y ** 2 * SIN(T) integrand 1
F2 = IINTEG (X, Y, F1)                          integral 1
F3 = RECONV (F2)                                display 1
```

Note that the parameters of the IINTEG operator are, from right to left, the integrand, the first (or only) variable of integration, and any other variables of integration in order. A single constant of integration is generated automatically as each indefinite integration is completed. If the integration can only be indicated, the constant of integration is omitted.

The case of definite integration is similar, except that the limits must be specified.

```
F10 = T ** 2                        a limit
F11 = A ** X * COS (Y)              integrand 2
F12 = DINTEG (A,H,X,F10,X,Y, F11)   integral 2
F13 = RECONV (F12)                  display 2
```

Note that the parameters for DINTEG consist of a triplet for each level of integration plus the integrand at the right. Each triplet has the lower limit at the left, the upper limit second, and the variable of integration at the right. In this example the first integration is with variable y, lower limit F10 (or t^2), and upper limit x. The second integration is with variable x, lower limit a, and upper limit h.

C_1	
$C_1 \cdot x^{-1}$	
$C_1 \cdot x^{C_2}$	$C_2 \neq -1$
$C_1 \cdot e^x$	
$C_1 \cdot C_2^{\,x}$	$C_2 \neq e$
$C_1 \cdot \sin(x)$	
$C_1 \cdot \cos(x)$	
$C_1 \cdot \sinh(x)$	
$C_1 \cdot \cosh(x)$	
$C_1 \cdot \text{loge}(x)$	
$\dfrac{C_1}{a + x^2}$	a = positive term
$\dfrac{C_1}{\sqrt{a - x^2}}$	a = positive term

Also any linear combination of the above (e.g., polynomials in x).

Note: C_1, C_2, a must not involve x, but need not be constants. All other integrations are merely indicated at present. This list may be extended for other particular cases.

Figure 9. Integrands Allowing Complete Integration in x.

```
F3
  X*CNST1
  +(1,5)*(X**5)*(Y**2)
  +(1,3)*(Y**3)*(LOGE(X))*(SIN(T))
  +CNST2
F13
  (A**(A))*((LOGE(A))**(-1))*(SIN(T**2))
  +(-1)*(A**(H))*((LOGE(A))**(-1))*(SIN(T**2))
  +DINTEG(A,H,X,((A**(X))*(SIN(X))))
```

Figure 10. Principal Output, Integration Examples

(In general the limits can be quite complex, but only a limited class of integrands can be integrated completely using SYMAP2; see Figure 9.)

In this example after one definite integration the integrand becomes $a^x \sin(x) - a^x \sin(t^2)$, a sum of terms. The first term cannot be integrated further by SYMAP2; so its second integration is merely indicated. The second term can be handled however, and its second integration is carried out. The result is shown in Figure 10.

As with most SYMAP2 operations the results, F2 and F12 here, can be used in further manipulations.

F. Symbolic Differentiation

Let us find the derivative of the algebraic form

$$a^{x \int_h^z \cos ty \, dt}$$

with respect to x and y and z separately. First we must generate this form unless we already have it from earlier steps:

F1 = T * Y	ty
F2 = COS (F1)	cos ty
F3 = DINTEG (H, Z, T, F2)	integral
F4 = X * F3	exponent
F5 = A ** F4	differand

We then differentiate the form F5 with respect to x in two steps:

```
F11 = DERIV (X, F5)
F12 = VALUE (F11)
```

Here F11 is the indicated derivative, not yet evaluated, and F12 is the result of evaluation, namely the L-form equivalent to

$$(a^{x \int_h^z \cos ty \, dt}) (\int_h^z \cos ty \, dt) (\log_e a).$$

To display this in readable form we use a RECONV step:

```
F20 = RECONV (F12)
```

(See Figure 11 and 12.)

```
EXACT
A
X
Y
Z
H
T
,,
GFCN
,,
F1 = T * Y
F2 = COS (F1)
F3 = DINTEG (H, Z, T, F2)
F4 = X * F3
F5 = A ** F4
F10 = RECONV (F5)
F11 = DERIV (X, F5)
F12 = VALUE (F11)
F20 = RECONV (F12)
,,
```

Figure 11. Input for Differentiation Example

```
F10
  A**(X*(DINTEG(H,Z,T,COS(Y*T))))
F20
  (A**(X*(DINTEG(H,Z,T,COS(Y*T)))))
  *(LOGE(A))*(DINTEG(H,Z,T,COS(Y*T)))
```

Figure 12. Principal Output, Differentiation Example

In like manner we can differentiate F5 with respect to y:

```
          F21 = DERIV (Y, F5)
          F22 = VALUE (F21)
          F30 = RECONV (F22)
```

Here we would get a FORTRAN-like expression readily interpreted as

$$\left(a^{x\int_h^z \cos ty\, dt}\right)\left(x\int_h^z (-\, t \sin ty)\, dt\right)(\log_e a)$$

The exact order of elements would depend heavily on the collating sequence used.

Three similar steps would get the derivative with respect to z as

$$\left(a^{x\int_h^z \cos ty\, dt}\right)(\, x \cos zy)\,(\log_e a).$$

Incidentally, the L-forms F12, F22, and the like can be differentiated further to obtain second- and higher-order derivatives if desired. Also numeric values can be substituted for some or all variables to obtain derivatives at specific points.

Many outside symbol manipulators such as ALTRAN[7] are designed to process polynomials or rational expressions only. An example like this one involving trigonometric functions and integrals could thus not be done with such manipulators.

G. Differentiation of a Function Defined Implicitly

Suppose that u is defined implicitly in terms of an independent variable x and two other variables v and w, which also depend on x, as follows:

$$F(x,u,v,w) = x^2 + u \cdot \log_e (x) + u^5 + v^3 \cdot w^2 = 0$$

where, say:

$$v = \sin (x)$$

$$w = x + \tan (x)$$

$$u = g (x)$$

Let us find the derivative of u with respect to x, that is $\frac{d}{dx} g(x)$, in terms of x, u, v, and w.

The SYMAP2 operator IMDIFF (implicit differentiation) is used to this end. Several preliminary steps are required, however:

```
     F1 = X **2  + U * LOGE(X) + U **5 + V **3 * W **2          F
     F2 = G(X)                                                  u
     F3 = SIN (X)                                               v
     F4 = X + TAN (X)                                           w
     F5 = IMDIFF (X, F1, U, F2, V, F3, W, F4)
     F6 = RECONV (F5)
```

The parameters of IMDIFF are, from left to right:

(1) The independent variable of differentiation, here x.
(2) The formula defining the dependent variable implicitly, here F.
(3) The dependent variable, here u.
(4) The general function (of x) representing the dependent variable, here g(x).
(5) One of the intermediate variables, if any, here v.
(6) The explicit definition of v in terms of x.
(7) Another intermediate variable, if any, here w.
(8) The explicit definition of w in terms of x.

Of course X, U, V, and W must be primitives (so declared in part (2) of the SYMAP2 input), and G must be a user specified function (as declared in part (3) of the input.)

The result of IMDIFF, stored with label F5 and displayed via the RECONV operator, is not the desired derivative $\frac{d}{dx} g(x)$ but is the total derivative of the expression F, which is equivalent to

$$\frac{\partial F}{\partial x} + \frac{\partial F}{\partial u} \cdot \frac{du}{dx} + \frac{\partial F}{\partial v} \cdot \frac{dv}{dx} + \frac{\partial F}{\partial w} \cdot \frac{dw}{dx} \qquad (=0)$$

since F is a function of x, u, v, and w in this example and u, v, and w are all functions of x.

In this example

$$\frac{\partial F}{\partial x} = 2.x + u.x^{-1}$$

$$\frac{\partial F}{\partial u} = \log_e(x) + 5.u^4 \qquad \frac{du}{dx} = \frac{dg(x)}{dx}$$

$$\frac{\partial F}{\partial v} = 3.\ v^2\ .\ w^2 \qquad \frac{dv}{dx} = \cos\ (x)$$

$$\frac{\partial F}{\partial w} = 2.v^3.w \qquad \frac{dw}{dx} = 1 + [\cos(x)]^{-2}$$

Thus F5 represents

$$(2x + u\ .\ x^{-1}) + (\log_e\ (x) + 5.u^4)\ .\ \frac{dg(x)}{dx} + 3.v^2.w^2.\ \cos(x)$$

$$+ 2.v^3.w.\ (1+ [\cos\ (x)\ ]^{-2}\ ) \qquad (=0)$$

which is a linear equation defining $\frac{dg(x)}{dx}$ implicitly. It can be solved using the SOLVEQ operator as follows:

F7 = DERIV (X, F2) $\qquad \frac{dg(x)}{dx}$

F8 = SOLVEQ (F5, F7, 1)

F9 = RECONV (F8)

The SOLVEQ parameter 1 indicates just one equation; so F5 and F7 serve as one-dimensional arrays without any need of the FSYST operator.

F8 represents the solution (a single element) and can be displayed directly without prior use of the ELEM operator. In this example the solution is the equivalent of

$$-\frac{(2.x + u.x^{-1} + 3.v^2.w^2.\ \cos(x) + 2.v^3.w + 2.v^3.w\ .\ [\cos(x)]^{-2})}{(\log_e\ (x) + 5\ .\ u^4)}$$

fully expanded as five terms, each containing the inverse of the denominator as a factor. This is the derivative wanted.

At present IMDIFF is limited to 0, 1, or at most 2 intermediate variables like v and w. If w had been missing from F in the example above the last two IMDIFF parameters could have been omitted. If no intermediate variable had been involved, the first four parameters would have been sufficient.

Note that if there were two or more independent variables, say x and y, the above procedure could be used to find either $\frac{\partial g\ (x,y)}{\partial y}$ or $\frac{\partial g\ (x,y)}{\partial x}$ separately. The only changes needed would be to express u, v, and w in terms of both x and y instead of x alone and to specify either x or y, but not both, as the first parameter of IMDIFF and of DERIV.

H. Differentiation of Two Functions Defined Implicitly

Suppose that u and v are defined implicitly in terms of an independent variable x by means of two equations, say:

$$F_1\ (x,u,v) = u\ .\ v^2 + c\ .\ \log_e\ (x) \qquad = 0$$

$$F_2\ (x,u,v) = u\ .\ \sin(x) - v\ .\ a^{2x} \qquad = 0$$

Also let

$$u = g_1(x)$$
$$v = g_2(x)$$

indicate the dependence of u,v on x.

Let us use SYMAP2 to find $\frac{du}{dx}$ and $\frac{dv}{dx}$, that is $\frac{dg_1(x)}{dx}$ and $\frac{dg_2(x)}{dx}$.

The operator IMDIFF cannot be used directly here, but another operator, PIMDIF, can. Because of the generality of PIMDIF several specifications and preliminary steps are needed:

```
F1 = U * V ** 2 + C * LOGE (X)
F2 = U * SIN (X) - V * A ** (2 * X)
F3 = FSYST (F1, F2)
```

This forms the system of equations defining the u and v implicitly. (In principle there could be several equations in several variables.)

F4 = G1 (X)	$u = g_1\ (x)$
F5 = G2 (X)	$v = g_2\ (x)$
F6 = SEQ (F4, F5)	sequence

These steps form an ordered sequence (technically different from a system in SYMAP2) of the functions showing a general dependence of u and v on x (or on several independent variables if that is appropriate).

```
        F7 = SEQ (X)                    independent
        F8 = SEQ (U, V)                 dependent
```

These two steps form ordered sequences of the independent variable(s) and of the dependent variables respectively. Finally,

```
        F9 = PIMDIF (1, 2, F7, F3, F8, F6)
```

There are always six parameters for PIMDIF, specified in this order:

(1) the number of independent variables
(2) the number of equations (and dependent variables)
(3) the sequence of independent variables (primitives)
(4) the system of equations
(5) the sequence of dependent variables (primitives)
(6) the sequence of dependency functions

Note that F7 here is the sequence containing x, not the primitive x itself.

The result F9 is a system of linear equations, here two equations in $\frac{dg_1(x)}{dx}$ and $\frac{dg_2(x)}{dx}$, which can be solved with SOLVEQ as follows:

```
        F11 = DERIV (X, F4)             unknown 1
        F12 = DERIV (X, F5)             unknown 2
        F13 = FSYST (F11, F12)          unknowns
        F14 = SOLVEQ (F9, F13, 2)       solutions
        F15 = ELEM (F14, 1)             solution 1
        F16 = ELEM (F14, 2)             solution 2
```

Thus F15 is the derivative $\frac{dg_1(x)}{dx}$ and F16 is $\frac{dg_2(x)}{dx}$, both expressed in terms of x, u, v. If desired they can be displayed using RECONV in the usual way.

If there had been two independent variables x and y and the sequences, functions, and numerical parameters of PIMDIF had been adjusted to show this, then the system F9 would have been two pairs of equations, with equations 1 and 2 in $\frac{\partial g_1(x,y)}{\partial x}$ and $\frac{\partial g_2(x,y)}{\partial x}$ and equations 3 and 4 in $\frac{\partial g_1(x,y)}{\partial y}$ and $\frac{\partial g_2(x,y)}{\partial y}$. A set of four equations in four unknowns is too large for SOLVEQ as implemented on BRLESC 2 at present; so two separate systems would have to be formed out of F9 using ELEM and FSYST. This could be done, however, with only a few extra SYMAP2 steps, and the two systems solved separately. Given a larger core memory on BRLESC 2, larger systems could be handled directly.

I. Change of Variables

Given a pair, say, of differential equations of the first order

$$x + y \cdot \frac{\partial g(x,y)}{\partial x} = 0$$

$$\sin(x) + \frac{\partial g(x,y)}{\partial y} = 0$$

let us change independent variables from (x,y) to (w,z) where

$$x = w + z$$

$$y = w \cdot z$$

The SYMAP2 operator CHGVAR was developed for this purpose. Several preliminary steps and specifications are required:

```
F1 = G(X,Y)
F2 = X + Y * DERIV (X, F1)
F3 = SIN (X) + DERIV (Y, F1)
F4 = FSYST (F2, F3)
```

This forms the system of differential equations to be transformed.

```
F5 = SEQ (X,Y)
F6 = SEQ (W,Z)
```

These form two ordered sequences of the old variables and of the new variables, respectively.

```
F7 = W + Z                   x(w,z)
F8 = W * Z                   y(w,z)
F9 = FSYST (F7, F8)
```

These state the relations of the old variables in terms of the new, and form a system of them (in the same order as for the sequence, namely first x and then y).

```
F11 = H1(X,Y)                w(x,y)
F12 = H2(X,Y)                z(x,y)
F13 = FSYST (F11, F12)
```

These state the general dependence of the new variables in terms of the old (using functions declared in part (3) of the input), and form a system (in the order w, then z).
Finally,

F14 = CHGVAR (2,2,2, F4, F5, F6, F9, F13)

There are always precisely eight parameters for CHGVAR, namely, from left to right:

(1)	the number of equations	q
(2)	the number of old variables	o
(3)	the number of new variables	$n \le o$
(4)	the system of equations	
(5)	the sequence of old variables	
(6)	the sequence of new variables	
(7)	the system, old in terms of new	
(8)	the system, new as functions of old	

The result of CHGVAR, here called F14, is a system of equations equivalent to the original but in terms of the new variables, namely the results wanted. They can be isolated using ELEM of course:

F15 = ELEM (F14, 1)
F16 = ELEM (F14, 2)

Finally they can be displayed if desired:

F17 = RECONV (F15)
F18 = RECONV (F16)

The new equations (see Figure 13) are equivalent to

$$(w-z)^{-1} . w . z^2 . \frac{\partial g}{\partial w} - (w-z)^{-1} . w.z^2 . \frac{\partial g}{\partial z} + w.z. \frac{\partial g}{\partial w} + w + z = 0$$

and

$$-(w-z)^{-1} . \frac{\partial g}{\partial w} + (w-z)^{-1} . \frac{\partial g}{\partial z} + \sin(w + z) = 0$$

Incidentally, the operator CHGVAR automatically makes repeated use of IMDIFF and PIMDIF, while PIMDIF also uses IMDIFF. This particular example was run on BRLESC 2 and required approximately 0.2 minute, including several displays. The time required depends on the particular relations, number of variables, number of derivatives present, etc.

```
Equations in (x,y) as displayed
  Y*(DERIV(X,G(X,Y)))+ X
  (DERIV(Y,G(X,Y)))+(SIN(X))

Equations in (w,z) where x = w + z, y = w . z
  ((W+(-1)*Z)**(-1))*W*Z**2*(DERIV(W,G(W,Z)))
  +(-1)*((W+(-1)*Z)**(-1))*W*Z**2*(DERIV(Z,G(W,Z)))
  +W*Z*(DERIV(W,G(W,Z)))
  +W
  +Z

  (-1)*((W+(-1)*Z)**(-1))*(DERIV(W,G(W,Z)))
  +((W+(-1)*Z)**(-1))*(DERIV(Z,G(W,Z)))
  +(SIN(W+Z))
```

Figure 13. Change of Variables Example

At present there are severe limitations on size in various parts of SYMAP2. For CHGVAR at most two old variables, at most two new variables, and at most three derivatives of first order can be handled. Some of these restrictions can be relaxed on special request.

J. Taylor's Series in Two Variables

For a more extensive application of SYMAP2 let us study in some detail how to expand a function of two variables in a Taylor's series about a point (a,b). The same technique would apply to any function f(x,y) but for definiteness let us consider f(x,y) = sin (x + y) and choose (a,b) = (0,0). For some other function f or some other point (a,b) only the first few manipulations, which specify these, need be changed; so the "program" is reusable in many Taylor's series contexts.

$$f(x,y) = f(a,b) + \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)f(x,y)\Big|_{(a,b)} + \frac{1}{2!}\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^2 f(x,y)\Big|_{(a,b)} + \dots + \frac{1}{n!}\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^n f(x,y)\Big|_{(a,b)} + Rn$$

Here $h = x - a$ and $k = y - b$ and the remainder Rn will be ignored. Thus repeated derivatives with respect to x and to y and mixed partials like $\frac{\partial^3 f}{\partial x^2 \partial y}$ will be needed. Further, the substitution of a for x and b for y must be done repeatedly. In general the higher derivatives of functions are much more complex than the functions themselves; so the character strings tend to get longer and some rather small finite n will have to be set to avoid overflow of memory blocs (whose size is preassigned via FORTRAN DIMENSION statements). This problem is less serious for sin (x + y) than for most functions.

We will make use of the control manipulations introduced earlier which allow counters and loops. (Without these many similar steps would be needed for the higher level terms, and additional steps would have to be added if n were increased.)

The flow diagram shown in Figure 14 will be used to clarify the SYMAP2 steps as they are introduced. Because of the looping involved, some strings need to be initialized prior to entering the loops, and some strings used repeatedly need to be formed early and saved for reuse. Some copying is needed also.

The i - th order terms in the Taylor series are given by:

$$\frac{1}{i\,!}\left(h \frac{\partial}{\partial x} + k \frac{\partial}{\partial y}\right)^i f(x,y) \Bigg|_{(a,b)}$$

$$= \frac{1}{i\,!}\sum_{j=0}^{i} \binom{i}{j} h^{i-j} k^{j} \frac{\partial^i f}{\partial x^{i-j}\,\partial y^j} \Bigg|_{(a,b)}$$

In the flow diagram boxes numbered 11 through 18 handle the variation of j from 1 through i and box 19 handles j = 0, the term involving $\frac{\partial^i f}{\partial x^i}$. Box 29 provides the variation of i up through n, looping back to box 10. As with any program, other sequences of operations could have been used to the same end.

Box 1 specifies the f(x,y) and the point (a,b), here (0,0), as well as the desired level of expansion n, say 3. We assume that

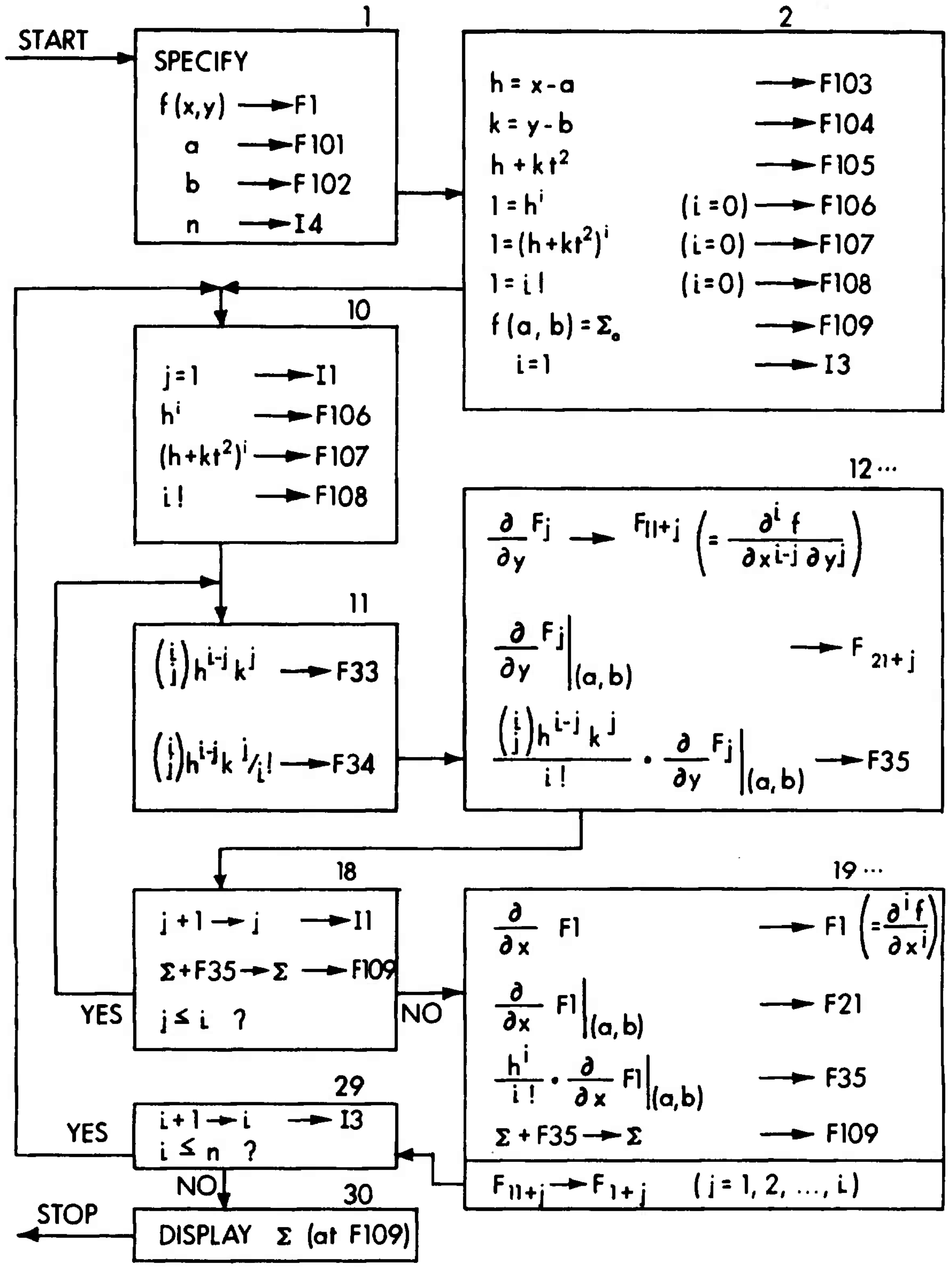

Figure 14. Flow Diagram, Taylor's Series Example.

primitives x, y, and t have been declared. The first SYMAP2 steps are:

```
        F101 = X + Y                    (x+y)
        F1   = SIN (F101)               f
```

These two basic manipulations specify f (x,y). (A single composite manipulation could have been used, if permitted.) If desired F1 can be displayed in readable form:

```
        F100 = RECONV (F1)
```

The point (a,b) = (0,0) is specified easily:

```
        F101 = 0                        a
        F102 = 0                        b
```

The level n is to be a small positive integer, say 3, in index I4. One step provides this:

```
        INDEX I4 = 3
```

This completes the variable part of the program. What follows could be the same for any reasonable f, any (a,b), and sufficiently small n.

Box 2 forms h = x - a and k = y - b, since various powers of these are needed and can be built up by repeated multiplication or by substitution in certain expressions involving them. The combination $h + kt^2$ raised to various powers can provide the binomial coefficients and related products such as $h^{i-j}k^j$ when needed in the loops. Thus we provide these SYMAP2 steps:

```
        F103 = X - F101                 x - a = h
        F104 = Y - F102                 y - b = k
        F105 = F104 * T                 kt
        F105 = F105 * T                 kt^2
        F105 = F105 + F103              h + kt^2
```

(The label F105 is reused here merely because the intermediate results are not needed further and there is a finite limit on the number of labels permitted.)*

The next few steps, as shown in box 2 of the flow diagram, set initial values prior to the iterative multiplications of box 10, which

*See Section VI

is within a loop with i varying.

```
F106 = 1                    h^i          ( i = 0 )
F107 = 1              (h + kt^2)^i       ( i = 0 )
F108 = 1                    i!           ( i = 0 )
```

Similarly the initial term of the sum which is the Taylor's series sought is f(a,b). This can be obtained from f(x,y) at F1 by successively substituting a for x and b for y:

```
F20 = SUBST (F1, X, F101)
F109= SUBST (F20, Y, F102)     f(a,b) = Σ (i=0)
```

This completes the initializations except for setting the index i to unity. This index i, kept at I3, refers to the current order of derivatives and will vary from 1 (since i = 0 has been taken care of) to n.

```
INDEX I3 = 1
```

We now consider the major loop shown in boxes 10 through 29, with i held fixed until near the end. The first step of box 10 must be a numbered manipulation since a jump is made to it from box 29. Let us use the number 10 and let box numbers and manipulation numbers agree, merely for convenience. (The first manipulations in boxes 1 and 2 and certain other boxes do not actually require numbers since there are no actual jumps to them. They are shown only for ease of reference.)

```
INDEX I1 = 1                10       numbered step
```

The above step sets index j, kept at I1, to unity. This index will vary from 1 through i for fixed i, and the case of j = 0 will be handled separately in box 19. Certain strings independent of j are best formed prior to entering the loops and hence are considered here:

```
F106 = F103 * F106                    h^i
F107 = F105 * F107              (h + kt^2)^i
INDEX F10 = I3                        i
F108 = F10 * F108                     i!
```

Note that F108 is an L-constant representing i!, as needed in all terms of level i. It is formed by multiplying the L-constant for i (at F10) by the previous F108, which was (i-1)! F107 is the expanded form of

$(h + kt^2)^i$ for the current i. F106 would also be a polynomial if a were not 0, but in this example is a simple power $h^i = (x - a)^i = (x - 0)^i = x^i$. Box 10 is now completed and the index j has been set to 1.

Since there is a jump to box 11 from box 18, the first step of box 11 must be a numbered one:

```
INDEX F10 = I1              11        numbered step
F10 = T ** F10                        t^j
F10 = F10 * F10                       t^2j
F31 = SUBST (F107, F10, 0)
F32 = F107 - F31
F33 = SUBST(F32, T, 1)
F34 = F33  /  F108
```

The above set of steps generate the needed "coefficient" of $\frac{\partial^i f}{\partial x^{i-j}\partial y^j}$, namely $\binom{i}{j} h^{i-j}k^j / i!$. Other steps could have been used, but this set accomplishes the goal as follows. F107 is $(h + kt^2)^i$ expanded, equivalent to $\sum_{m=0}^{i}\binom{i}{m} h^{i-m}k^m t^{2m}$. It contains terms in $t^2, t^4, \ldots, t^{2i}$ and terms not involving t (m=0). We want to isolate the terms in t^{2j}, that is those terms for which m = j, and thus obtain $\binom{i}{j} h^{i-j}k^j t^{2j}$. By generating t^{2j} at F10, using the first 3 steps of box 11 as shown above, and then replacing this power (only) by 0 in F107, we obtain at F31 all the terms of F107 <u>except</u> the ones wanted. Then subtracting F31 from F107 gives at F32 exactly the terms wanted. Replacing t (and indeed <u>all</u> powers of t) by 1 in F32 then gives at F33 the expression $\binom{i}{j} \overline{h^{i-j}k^j}$. (If a or b is not zero, this is a polynomial in x, y, a, b; otherwise one term.) Finally since F108 contains i! (see box 10), division of F33 by F108 gives at F34 the "coefficient" wanted.

We next evaluate the appropriate derivative $\frac{\partial^i f}{\partial x^{i-j}\partial y^j}$ for the current fixed i and fixed j. One way to do this is to use slightly different notation and a separate set of steps for each j. Let us introduce box 12 for j=1, box 13 for j=2, etc., as shown on the expanded flow diagram detail of Figure 15.

Applying this approach we continue box 11 with a series of decisions on j, up to the allowed maximum that n can have:

```
        INDEX I2 = I1 - I4
        INDEX I2 = I2 - 1
        IF (I2, 19)                          j > n ?
        INDEX I2 = 1 - I1
        IF (I2, 12)                          j = 1 ?
        INDEX I2 = 2 - I1
        IF (I2, 13)                          j = 2 ?
        INDEX I2 = 3 - I1
        IF (I2, 14)                          j = 3 ?
        .
        .
        .
        GO TO 19                             (j > max n)
```

A numbered step starts box 12:

```
        F10 = DERIV (Y, F1)             12   numbered step
        F12 = VALUE (F10)
```

$$\frac{\partial}{\partial y} \mathrm{F1} = \frac{\partial^i f}{\partial x^{i-1} \partial y}$$

```
        F20 = SUBST (F12, X, F101)
        F22 = SUBST (F20, Y, F102)           at (a,b)
        F35 = F22 * F34                      times coefficient
        GO TO 18
```

Similarly for box 13:

```
        F10 = DERIV (Y, F2)             13   numbered step
        F13 = VALUE (F10)
```

$$\frac{\partial}{\partial y} \mathrm{F2} = \frac{\partial^i f}{\partial x^{i-2} \partial y^2}$$

```
        F20 = SUBST (F13, X, F101)
        F23 = SUBST (F20, Y, F102)           at (a, b)
        F35 = F23 * F34                      times coefficient
        GO TO 18
```

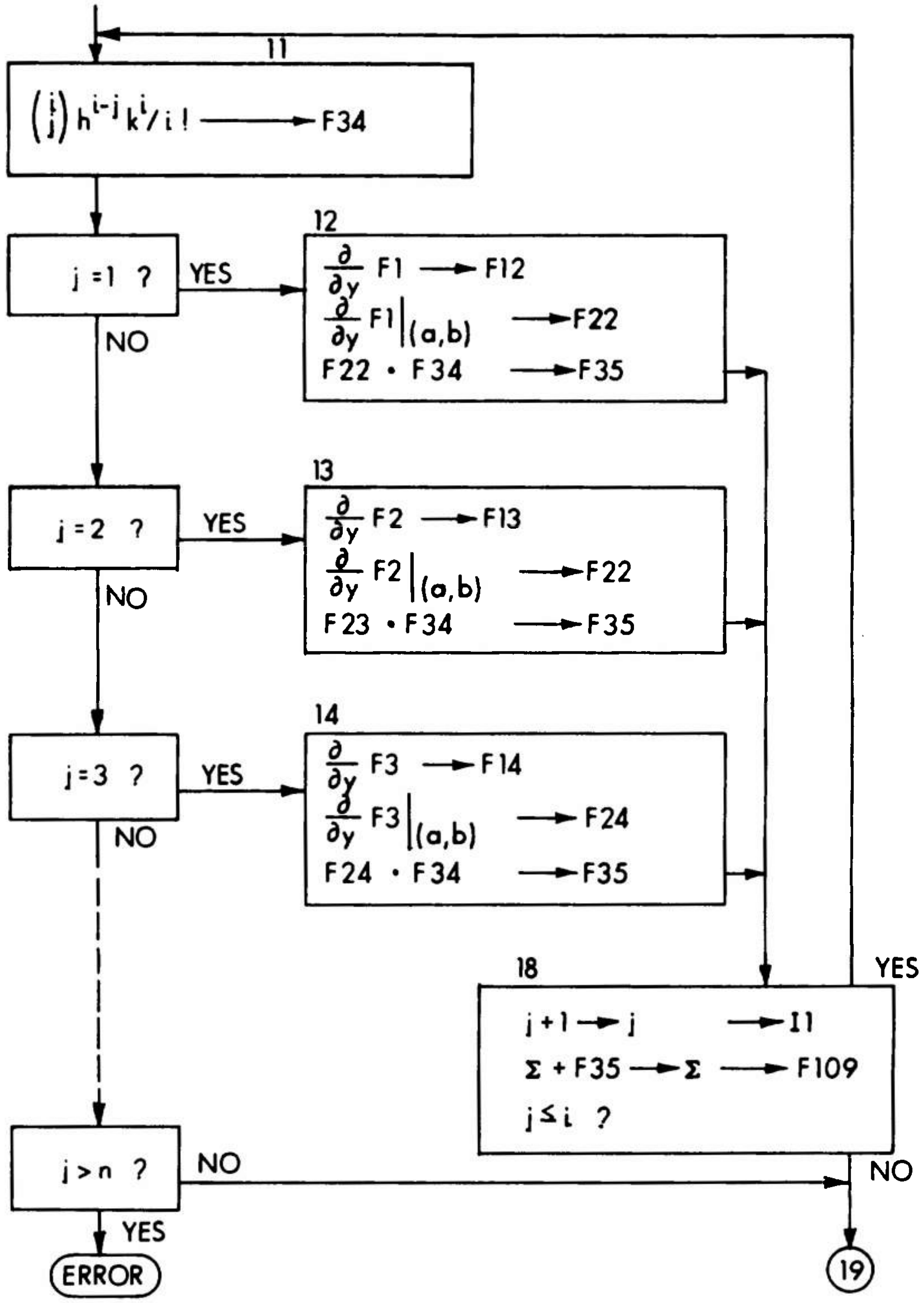
11
$\binom{i}{j} h^{i-j} k^{i} / i! \longrightarrow F34$
j = 1 ?
YES
NO
12
$\frac{\partial}{\partial y} F1 \longrightarrow F12$
$\frac{\partial}{\partial y} F1 \big|_{(a,b)} \longrightarrow F22$
$F22 \cdot F34 \longrightarrow F35$
j = 2 ?
YES
NO
13
$\frac{\partial}{\partial y} F2 \longrightarrow F13$
$\frac{\partial}{\partial y} F2 \big|_{(a,b)} \longrightarrow F22$
$F23 \cdot F34 \longrightarrow F35$
j = 3 ?
YES
NO
14
$\frac{\partial}{\partial y} F3 \longrightarrow F14$
$\frac{\partial}{\partial y} F3 \big|_{(a,b)} \longrightarrow F24$
$F24 \cdot F34 \longrightarrow F35$
18
$j+1 \longrightarrow j \longrightarrow I1$
$\Sigma + F35 \longrightarrow \Sigma \longrightarrow F109$
$j \leq i$?
YES
NO
j > n ?
NO
YES
ERROR
19

Figure 15. Detail of Boxes 11 through 18, Taylor's Series

Analogous steps apply for box 14, etc, for each allowed j. Each of these boxes leads to box 18:

```
          INDEX I1 = I1 + 1            18   numbered step
          F109 = F35 + F109                 enlarged Σ
          INDEX I2 = I3 - I1
          IF (I2, 11)                       j ≤ i ?
```

The above set of steps increases j, accumulates terms into the series, and if j ≤ i loops back to box 11 to continue with the new j. Otherwise box 19 is done next:

```
          F10 = DERIV (X, F1)          19   numbered step
          F1  = VALUE (F10)                 $\frac{\partial}{\partial x} F1 = \frac{\partial^i f}{\partial x^i}$
          F20 = SUBST (F1, X, F101)
          F21 = SUBST (F20, Y, F102)        at (a, b)
```

Note that in this case the derivative is taken with respect to x, not y as in boxes 12, 13, etc. Also a different "coefficient", namely $h^i/i!$, is needed:

```
          F34 = F106 / F108                 $h^i/i!$
          F35 = F34 * F21                   new "term"
          F109 = F35 + F109                 enlarged Σ
```

This completes the accumulation of terms at level i, but some adjustment is needed before the next i can be handled. The i-th level derivatives are not at F1, F2, F3, etc. as needed in box 12, 13 14, etc. but rather at F1, F12, F13, etc. F1 is correct because box 19 was done <u>last</u> at level i and the previous F1 was no longer needed. F12 was used in box 12 because the old F2 was still needed for box 13, and similarly for the others. Hence, these must now be copied into place, say with a loop on j = 2,3, . . . , i as follows:

```
          INDEX I1 = 2                      j = 2
          INDEX I2 = 2 - I1            21   numbered step
          IF (I2, 22)                       j = 2 ?
          INDEX I2 = 3 - I1
          IF (I2, 23)                       j = 3 ?
          INDEX I2 = 4 - I1
          IF (I2, 24)                            j = 4 ?
          .
          .
          .
          GO TO 29
          F2 = F12                          22
          GO TO 28
          F3 = F13                          23
          GO TO 28
          F4 = F14                          24
          GO TO 28
          INDEX I1 = I1 + 1                 28   j + 1 → j
          INDEX I2 = I3 - I1
          IF (I2, 21)                            j ≤ i ?
```

(Doing the above by a loop is not strictly necessary here, but in another context this technique might be very useful.)

We are now ready to advance the index i and loop back to box 10 if required:

```
          INDEX I3 = I3 + 1                 29   i + 1 → i
          INDEX I2 = I4 - I3
          IF (I2, 10)                            i ≤ n ?
```

If I2 is negative, we have the entire Taylor's series at F109 and can display it in readable form:

```
          F200 = RECONV (F109)
```

The original expression for f(x,y) was destroyed because we reused label F1 at each level i. If we had wanted to save it, we could have copied it from F1 to another location in box 1.

The original and final displays are shown in Figure 16.

The above set of expository examples has demonstrated some, but by no means all, of the capability of SYMAP2. The reader is urged to consider ways it can be of use to him.

```
F100
  (SIN(X+Y))
F200
  X
  +(1,2)*X*(Y**2)
  +(1,2)*(X**2)*Y
  +(1,6)*(X**3)
  +Y
  +(1,6)*(Y**3)
```

Figure 16. Principal Output, Taylor's Series Example

VI. PRESENT STATUS OF SYMAP2 AND FUTURE PLANS

The algebraic symbol manipulator SYMAP2 is still experimental though operational and is continuously being modified. It has been used for a variety of small applications and a few large ones within BRL, however; and additional applications are welcome. Local and other potential users are encouraged to contact the author.

On the BRLESC 2 computer 80000 words of core memory are required at present for efficient processing of 300-character strings, and longer symbol strings require more storage than this. Some temporary use of disc storage as well as 96000 words of core permits symbol strings up to 700 characters in length at a cost of greatly increased running time for disc accesses and exchanges. A much larger core or virtual memory for BRLESC 2 or its successor is needed for larger problems requiring strings of 1000 to 2000 characters.

Restrictions on user "programs" of manipulations include the following at present (96000 word core memory):

(1) Not more than 40 primitives specified.

(2) Not more than 30 special user functions specified.

(3) Not more than 200 distinct names of results Fi (but names can be reused if desired).

(4) Not more than 700 characters in any string, including expanded unsimplified intermediate results.

(5) Not more than 11000 characters total in all results saved.

(6) Not more than 9 distinct indexes In used in arrays (but these indexes can be used repeatedly).

(7) Not more than 20 distinct counters In (reusable).

(8) Not more than 70 characters per manipulation specification.

(9) Not more than 200 basic manipulations per program (steps within loops being counted only once).

(10) Not more than 3 simple linear equations in a system to be solved.

A variety of error prints normally inform the user when such restrictions are not adhered to. However, some of these restrictions can be relaxed somewhat, at the expense of others, by special arrangement. Most can be relaxed if additional memory becomes available.

Future extensions of SYMAP2 will probably be in these directions:

(1) Additional standard functions if needed.

(2) Simpler user commands to replace frequently needed combinations of present basic manipulations.

(3) More options to permit greater user control.

(4) Improved techniques for reuse of storage if released.

(5) An on-line version for use in remote-access or time-sharing environments if sufficient storage becomes available.

(6) More "natural" displays, with raised exponents, for example.

(7) Fewer restrictions on labels, to allow applications oriented names.

(8) Alternative versions of SYMAP2 to respond to special needs.

REFERENCES

1. L. Fox, ed., Advances in Programming and Non-Numerical Computation, Pergamon Press, New York, 1966.
2. Peter J. Smith, "Symbolic Derivatives Without List Processing, Subroutines, or Recursion," BRL Memorandum Report No. 1630, Feb. 1965 (AD 465 252). Also in Communications of the ACM, Vol. 8, No. 8, August 1965, pp. 494-496.
3. George C. Francis, "SYMAP1 - An Experimental Symbol Manipulation Program," BRL Memorandum Report No. 2060, August 1970 (AD 712 996).
4. Daniel G. Bobrow, ed., Symbol Manipulation Languages and Techniques, North-Holland Publishing Company, Amsterdam, 1968.
5. Jean E. Sammet, "An Annotated Descriptor Based Bibliography on the Use of Computers for Non-Numerical Mathematics," Computing Reviews 7, No. 4, July-August 1966.
6. Jean E. Sammet, "An Overall View of FORMAC," IBM Technical Report No. TR00.1367, November 1965.
7. Andrew D. Hall, Jr., "The ALTRAN System for Rational Function Manipulation - A Survey," Communications of the ACM, Vol. 14, No. 8, August 1971.
8. Ceslovas Masaitis, Georege C. Francis, and Viola Woodward, "Symbolic Calculation by Computer," BRL Report (to appear).
9. George C. Francis and Viola Woodward, "Designing SYMAP2, An Algebraic Symbol Manipulation Program," BRL Memorandum Report (to appear).
10. Lloyd W. Campbell and Glenn A Beck, "BRLESC I/II FORTRAN," ARDC Technical Report No. 5, March 1970 (AD 704 343).

DISTRIBUTION LIST

NO. OF COPIES — ORGANIZATION

12 COMMANDER
DEFENSE DOCUMENTATION CENTER
ATTN TIPCR
CAMERON STATION
ALEXANDRIA, VA 22314

1 DIRECTOR
ADVANCED RESEARCH PROJECTS AGENCY
DEPARTMENT OF DEFENSE
WASH DC 20301

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCDL
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCRD
DR. J.V.R. KAUFMAN
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCRD-BN
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCRD-M
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCRD-TE
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCRD-TP
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCCP
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY MATERIEL COMMAND
ATTN AMCPM-SA
MR. WILKINSON
WASH DC 20315

1 COMMANDING GENERAL
U.S. ARMY AVIATION SYSTEMS COMMAND
ATTN AMSAV-E
12TH AND SPRUCE STREETS
ST LOUIS, MO 63166

1 COMMANDING GENERAL
U.S. ARMY ELECTRONICS COMMAND
ATTN AMSEL-CE
FORT MONMOUTH, NJ 07703

3 COMMANDING GENERAL
U.S. ARMY MISSILE COMMAND
ATTN AMSMI-AML
AMSMI-RF
AMSMI-RFC
REDSTONE ARSENAL, AL 35809

3 COMMANDING GENERAL
U.S. ARMY MISSILE COMMAND
ATTN AMSMI-R
AMCPM-LC
AMCPM-PE
REDSTONE ARSENAL, AL 35809

1 COMMANDING GENERAL
U.S. ARMY TANK-AUTOMOTIVE COMMAND
ATTN AMSTA-RHFL
WARREN, MI 48090

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
2	COMMANDING OFFICER U.S. ARMY MOBILITY EQUIPMENT RESEARCH AND DEVELOPMENT CENTER ATTN TECH DOCU CEN, BLDG 315 AMSME-RZT FORT BELVOIR, VA 22060
1	COMMANDING GENERAL U.S. ARMY MUNITIONS COMMAND ATTN AMSMU-RE DOVER NJ 07801
1	DIRECTOR U.S. ARMY MUCOM OPERATIONS RESEARCH GROUP EDGEWOOD ARSENAL, MD 21010
3	COMMANDING OFFICER U.S. ARMY EDGEWOOD ARSENAL ATTN SMUEA-R SMUEA-TSTI-L SMUEA-BL-S, W. SACCO EDGEWOOD ARSENAL, MD 21010
1	COMMANDING OFFICER U.S. ARMY FRANKFORD ARSENAL ATTN SMUFA-L1C00, S. EISMAN PHILADELPHIA, PA 19137
5	COMMANDING OFFICER U.S. ARMY PICATINNY ARSENAL ATTN SMUPA-DV SMUPA-TT SMUPA-TW SMUPA-VA6 SMUPA-VE DOVER, NJ 07801
4	COMMANDING OFFICER U.S. ARMY PICATINNY ARSENAL ATTN SMUPA-DW6 SMUPA-CR3 SMUPA-DS2 SMUPA-CB DOVER, NJ 07801
2	COMMANDING OFFICER U.S. ARMY PICATINNY ARSENAL ATTN SMUPA-V SMUPA-D DOVER, NJ 07801
1	COMMANDING GENERAL U.S. ARMY WHITE SANDS MISSILE RANGE ATTN STEWS-TE-E, MR. ELDER WHITE SANDS MISSILE RANGE, NEW MEXICO 88002
4	COMMANDING GENERAL U.S. ARMY WEAPONS COMMAND ATTN AMSWE-RE AMSWE-RDA AMSWE-RDF FLD SVC DIV ROCK ISLAND, IL 61202
1	COMMANDING OFFICER U.S. ARMY ROCK ISLAND ARSENAL ROCK ISLAND, IL 61202
1	COMMANDING OFFICER U.S. ARMY WATERVLIET ARSENAL WATERVLIET, NY 12189

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
2	COMMANDING OFFICER U.S. ARMY HARRY DIAMOND LABORATORIES ATTN LIB AMXDO-TD/002 WASH DC 20438
1	COMMANDING OFFICER U.S. ARMY MATERIALS AND MECHANICS RESEARCH CENTER ATTN AMXMR-ATL WATERTOWN, MA 02172
1	DIRECTOR U.S. ARMY ADVANCED MATERIEL CONCEPTS AGENCY 2461 EISENHOWER AVENUE ALEXANDRIA, VA 22314
1	COMMANDING GENERAL U.S. ARMY NATICK LABORATORIES ATTN AMXRE, DR. D. SIELING NATICK, MA 01762
1	COMMANDING OFFICER U.S. ARMY FOREIGN SCIENCE AND TECHNOLOGY CENTER FEDERAL OFFICE BLDG 220 7TH STREET, NE CHARLOTTESVILLE, VA 22901
1	DIRECTOR U.S. ARMY AIR MOBILITY RESEARCH AND DEVELOPMENT LABORATORY AMES RESEARCH CENTER MOFFETT FIELD, CA 94035
1	DIRECTOR U.S. ARMY AERONAUTICAL RESEARCH LABORATORY MOFFETT NAVAL AIR STATION, CALIFORNIA 94035
5	COMMANDING GENERAL U.S. ARMY COMBAT DEVELOPMENTS COMMAND ATTN CDCMR-I CDCMR-W CDCMR-U CDCMR-P (2 CYS) FORT BELVOIR, VA 22060
1	COMMANDING GENERAL U.S. ARMY COMBAT DEVELOPMENTS COMMAND STRATEGIC STUDIES INSTITUTE CARLISLE BARRACK, PA 17013
1	COMMANDING GENERAL U.S. ARMY COMBAT DEVELOPMENTS COMMAND COMBAT SYSTEMS GROUP FORT LEAVENWORTH, KS 6602
1	COMMANDING GENERAL U.S. ARMY COMBAT DEVELOPMENTS COMMAND EXPERIMENTATION COMMAND FORT ORD, CA 93941
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND INSTITUTE OF SPECIAL STUDIES FORT BELVOIR, VA 22060

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND NUCLEAR AGENCY FORT BLISS, TX 79916
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND AIR DEFENSE AGENCY ATTN MR. J. LAGOUROS FORT BLISS, TX 79916
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND ARMOR AGENCY ATTN CAGAR-PC FORT KNOX, KY 40121
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND FIELD ARTILLERY AGENCY FORT SILL, OK 73504
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND CHEMICAL-BIOLOGICAL- RADIOLOGICAL AGENCY FORT MCCLELLAN, AL 36205
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND ENGINEER AGENCY FORT BELVOIR, VA 22060
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND INFANTRY AGENCY FORT BENNING, GA 31905
1	COMMANDING OFFICER U.S. ARMY COMBAT DEVELOPMENTS COMMAND TRANSPORTATION AGENCY FORT EUSTIS, VA 23604
1	COMMANDING GENERAL U.S. ARMY COMBAT DEVELOPMENTS COMMAND PERSONNEL AND LOGISTICS SYSTEMS GROUP FORT LEE, VA 23801
3	COMMANDANT U.S. ARMY ARTILLERY AND MISSILE SCHOOL ATTN AKPSIAS-G AKPSIAS-CA AKPSIAS-G-RK FORT SILL, OK 73504
1	HQ DA (DACS-CW) WASH DC 20310
1	HQ DA (DAFD) WASH DC 20310
1	HQ DA (DAFD-ZAA) MR. A. GOLUB WASH DC 20310
1	HQ DA (DALO) WASH DC 20310
1	HQ DA (DARD-DDC) WASH DC 20310
1	HQ DA (DARD-MSN) WASH DC 20310
1	HQ DA (DARD-ARS) WASH DC 20310
1	HQ DA (DARD-ARP-M) WASH DC 20310

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
1	COMMANDING GENERAL U.S. ARMY TOPOGRAPHIC COMMAND ATTN TPC (14200) 6500 BROOKS LANE WASH DC 20315
1	COMMANDING OFFICER U.S. ARMY ENGINEER WATERWAYS EXPERIMENTATION AGENCY ATTN DR. A. SAKURAI P.O. BOX 631 VICKSBURG, MS 39181
2	DIRECTOR U.S. ARMY RESEARCH OFFICE ATTN DR. I. HERSHNER MR. M. WEIK, JR. 3045 COLUMBIA PIKE ARLINGTON, VA 22204
1	COMMANDING OFFICER U.S. ARMY RESEARCH OFFICE (DURHAM) ATTN DR. A. GALBRAITH BOX CM, DUKE STATION DURHAM, NC 27706
1	DEPARTMENT OF ORDNANCE U.S. MILITARY ACADEMY ATTN ASSOC PROF, MR. BURTON WEST POINT, NY 10996
1	COMMANDANT U.S. ARMY WAR COLLEGE CARLISLE BARRACK, PA 17013
1	COMMANDANT U.S. ARMY COMMAND AND GENERAL STAFF COLLEGE FORT LEAVENWORTH, KS 6602
1	CHIEF, U.S. ARMY STRATEGY AND TACTICS ANALYSIS GROUP 8120 WOODMONT AVENUE BETHESDA, MD 20014
1	MATHEMATICS RESEARCH CENTER U.S. ARMY UNIVERSITY OF WISCONSIN ATTN DR. J. ROSSER MADISON, WI 53706
3	COMMANDER U.S. NAVAL AIR SYSTEMS COMMAND ATTN AIR-604 WASH DC 20360
3	COMMANDER U.S. NAVAL ORDNANCE SYSTEMS COMMAND ATTN ORD-9132 WASH DC 20360
1	COMMANDER U.S. NAVAL MISSILE CENTER POINT MUGU CA 93041
1	COMMANDER U.S. NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER WASH DC 20007
1	COMMANDER U.S. NAVAL ORDNANCE LABORATORY ATTN DR. J. W. ENIG SILVER SPRING MD 20910
2	DIRECTOR U.S. NAVAL RESEARCH LABORATORY ATTN CODE 7600, TECH LIB WASH DC 20390

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
1	COMMANDER U.S. NAVAL WEAPONS LABORATORY DAHLGREN, VA 22448
1	AFATL (DLRV) EGLIN AFB, FL 32542
1	AFATL (DLRD) EGLIN AFB, FL 32542
1	AFATL (DLR) EGLIN AFB, FL 32542
1	AFWL (WLL) KIRTLAND AFB, NM 87117
1	DIRECTOR NATIONAL BUREAU OF STANDARDS ATTN DR. W. H. PELL DEPARTMENT OF COMMERCE WASH DC 20234
1	HEADQUARTERS U.S. ATOMIC ENERGY COMMISSION ATTN TECH LIB WASH DC 20545
2	DIRECTOR LAWRENCE LIVERMORE LABORATORY ATTN DR. W. NOH MR. M. WILKINS P.O. BOX 808 LIVERMORE, CA 94550
1	DIRECTOR LOS ALAMOS SCIENTIFIC LABORATORY P.O. BOX 1663 LOS ALAMOS, NM 87544
1	DIRECTOR NASA SCIENTIFIC AND TECHNICAL INFORMATION FACILITY ATTN SAK/DL P.O. BOX 33 COLLEGE PARK, MD 20740
1	DIRECTOR NATIONAL AERONAUTICS AND SPACE ADMINISTRATION LANGLEY RESEARCH CENTER LANGLEY STATION HAMPTON, VA 23365
1	DIRECTOR NATIONAL AERONAUTICS AND SPACE ADMINISTRATION LEWIS RESEARCH CENTER 21000 BROOKPARK ROAD CLEVELAND, OH 44135
1	AVCO CORPORATION RESEARCH AND ADVANCED DEVELOPMENT DIVISION 201 LOWELL STREET WILMINGTON, MA 01887
1	BATTELLE MEMORIAL INSTITUTE ATTN MR. FRED TIETZEL (STOIAC) 505 KING AVE. COLUMBUS, OH 43201
2	FIRESTONE TIRE AND RUBBER COMPANY ATTN LIB MR. M. C. COX 1200 FIRESTONE PARKWAY AKRON, OH 44317

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
2	GENERAL MOTORS CORPORATION DEFENSE RESEARCH LABORATORIES ATTN MR. J. GEHRING DR. A. CHARTERS SANTA BARBARA, CA 93108
1	THE MARTIN-MARIETTA CORPORATION ATTN TECH LIB BALTIMORE, MD 21203
1	THE MARTIN-MARIETTA CORPORATION AEROSPACE DIVISION ORLANDO, FL 32805
1	THE RAND CORPORATION 1700 MAIN STREET SANTA MONICA, CA 90406
1	SANDIA CORPORATION ATTN INFO DIST DIV P.O. BOX 5800 ALBURQUERQUE, NM 87115
1	SHOCK HYDRODYNAMICS, INC. ATTN DR. L. ZERNOW 15010 VENTURA BOULEVARD SHERMAN OAKS, CA 91403
1	UNITED AIRCRAFT CORPORATION MISSILES AND SPACE SYSTEMS GROUP HAMILTON STANDARD DIVISION WINDSOR LOCKS, CT 06096
1	BROWN UNIVERSTIY DIVISION OF ENGINEERING PROVIDENCE, RI 02912
1	CARNEGIE MELLON UNIVERSITY DEPARTMENT OF PHYSICS ATTN PROF. E. M. PUGH PITTSBURGH PA 15213
1	CORNELL UNIVERSITY ATTN PROFESSOR G. LUDFORD ITHACA, NY 14850
1	DREXEL INSTITUTE OF TECHNOLOGY DEPARTMENT OF AERONAUTICAL ENGINEERING ATTN PROF PEI CHI CHOU 32ND AND CHESTNUT STS. PHILADELPHIA, PA 19104
2	HARVARD UNIVERSTIY ATTN PROF. G. BIRKHOFF PROF. G. CARRIER CAMBRIDGE, MA 02139
1	IIT RESEARCH INSTITUTE ATTN LIB 10 WEST 35TH STREET CHICAGO, IL 60616
1	JOHNS HOPKINS UNIVERSITY ATTN PROF. C. TRUESDELL 34TH AND CHARLES STS. BALTIMORE, MD 21218
1	LINCOLN LABORATORY (MIT) 244 WOOD STREET LEXINGTON, MA 02173
1	NEW YORK UNIVERSITY - COURANT INSTITUTE ATTN PROF. P. GARABEDIAN NEW YORK, NY 10012

DISTRIBUTION LIST

NO. OF COPIES	ORGANIZATION
1	PENNSYLVANIA STATE UNIVERSITY ENGINEERING MECHANICS DEPARTMENT ATTN PROF N. DAVIDS UNIVERSTIY PARK, PA 16802
2	STANFORD RESEARCH INSTITUTE ATTN DR. M. COWPERTHWAITE DR. M. EVANS 333 RAVENWOOD AVENUE MENLO PARK, CA 94025
1	STEVENS INSTITUTE OF TECHNOLOGY DAVIDSON LABORATORY ATTN DR. C. GROSCH HOBOKEN, NJ 07030
1	UNIVERSITY OF CALIFORNIA ATTN PROF. M. PROTTER BERKELEY, CA 94704
1	UNIVERSITY OF CALIFORNIA AERONAUTICAL SCIENCES DIVISION ATTN PROF. M. HOLT BERKELEY, CA 94704
1	UNIVERSITY OF DELAWARE ATTN PROF. W. AMES NEWARK, DE 19711
1	UNIVERSITY OF MARYLAND ATTN PROF. M. MARTIN COLLEGE PARK, MD 20740
1	UNIVERSITY OF UTAH HIGH VELOCITY LABORATORY SALT LAKE CITY, UT 84112
1	UNIVERSITY OF WISCONSIN ATTN PROF. R. GUNDERSON MILWAUKEE, WI 53200
	ABERDEEN PROVING GROUND -------- ------- ------
	CH, TECH LIB MARINE CORPS LN OFC CDC LN OFC TECH DIR., USAHEL

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* U.S. Army Aberdeen Research and Development Center, Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

SYMAP2- An Operational Computer-Based Algebraic Symbol Manipulator

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

George C. Francis

6. REPORT DATE June 1972	7a. TOTAL NO. OF PAGES 66	7b. NO. OF REFS 10

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. RDT&E No. 1T061102A14B	BRL Memorandum Report No. 2199
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY US Army Materiel Command Washington, D. C.

13. ABSTRACT

SYMAP2, a FORTRAN computer program for symbolic manipulation of algebraic forms, is operational on the BRLESC 2 computer. FORTRAN-like formulas can be built up, combined, displayed, differentiated, modified through substitution or change of variables, and operated on in various other ways, either to verify or eliminate the need for many tedious and lengthy hand transformations. No knowledge of FORTRAN programming as such is required for use of SYMAP2. This report includes numerous examples with detailed explanation of each.

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Symbol manipulation						
Algebraic forms						
Functions of several variables						
Rational or approximate arithmetic						
Substitution						
Automatic simplifications						
Symbolic differentiation						
Symbolic integration						
Expansion of sums or products						
Solution of linear equations						
Change of variables						
Masaitis L-System						
Non-numeric computer applications						
BRLESC 2 computer program						

Unclassified
Security Classification